SelfLinux-0.13.1

Software-Installation

Autor: Oliver Boehm (*boehm@2xp.de*)
Autor: Mirko Zeibig (*mirko-lists@zeibig.net*)
Formatierung: Matthias Hagedorn (*matthias.hagedorn@selflinux.org*)

Inhaltsverzeichnis

1 Installieren und Deinstallieren von Software

2 Linux-Distributoren

3 RPM

3.1 Kompilieren von Source-RPMs
3.2 Anfragen der RPM-Datenbank
3.3 Graphische RPM-Frontends

4 Debian Paket Format

5 Die klassische Installation

6 Perl-Archive

7 Selbstauspackende Archive

8 Software-Archive

1 Installieren und Deinstallieren von Software

Leider gibt es noch kein einheitliches Installations-Verfahren unter Linux, aber mit dem RPM (Redhat Package Manager}-Format von *RedHat* hat sich mittlerweile ein Format durchgesetzt, das auch von anderen Distributoren verwendet wird. Daneben gibt es noch andere Formate und Verfahren, von denen die Gebräuchlichsten hier vorgestellt werden.

Eines kennzeichnet aber sämtliche Installationsverfahren: kein Reboot nach erfolgter Installation. Sämtliche Tools können sofort gestartet werden, evtl. ist das Starten eines Dienstes (Unix-Jargon: Daemon) nötig, was von der Kommandozeile aus erfolgt.

2 Linux-Distributoren

Die meisten Linux-Distributionen sind recht umfangreich und enthalten bereits die nötigen Tools und Programme, die bei der Grund-Installation ausgewählt wurden. Will man später das eine oder andere Programm nachträglich installieren, kann dies mit den Distributions-eigenen Werkzeugen erfolgen. Auch die Deinstallation ist über diesen Weg möglich. Voraussetzung dafür ist, dass das gewünschte Programm im Distributions-Umfang mit dabei ist.

Leider liegen manche Programme nicht in der neuesten Version vor, andere Programme fehlen, weil beispielsweise die Lizenz des Herstellers nicht mit der Distribution vereinbar ist. Dann muss man sich selbst darum kümmern, an die aktuelle Version des gewünschten Paketes zu kommen, um diese auf dem Rechner installieren zu können.

3 RPM

Das RPM-Format, das von *RedHat* für ihre Distribution entwickelt wurde, enthält zusammen mit einigen Verwaltungsdaten das compilierte Programm-Paket. Erkennbar sind RPM-Dateien an der Endung .rpm, wobei zusätzlich die Architektur (z. B. **i386** oder **alpha**) im Namen der Datei enthalten ist. So kennzeichnet

kaffe-1.0.6-2.i386.rpm

das Kaffe-Paket für die Intel386-Architektur. Pakete, die nicht an eine bestimmte Architektur gebunden sind (z. B. manche Java-Pakete) erhalten die Endung .noarch.rpm. Handelt es sich um ein Paket in Source-Form, so wird dies durch .src.rpm gekennzeichnet.

Folgende Eigenschaften kennzeichnen das RPM-Format:

* Prüfung, ob die Voraussetzung für ein Paket vorhanden ist
* lokale Installation
* Installation per FTP möglich
* Deinstallation

Wer über **FTP** installieren will, kann als Paket-Name eine **URL** angeben, z. B.

```
user@linux ~/ $ rpm -ih
ftp://ftp.redhat.com/pub/redhat/i386/RedHat/RPMS/kaffe-1.0.6-2.i386.rpm
```

Das Schöne an der Installation per **FTP** ist, dass die Abhängigkeiten vor der eigentlichen Installation überprüft werden, d. h. das restliche Paket wird erst heruntergeladen, wenn die Abhängigkeiten erfüllt sind. Dazu teilt sich der eigentliche Installations-Vorgang in drei Phasen auf:

1. das Pre-Install-Skript wird ausgeführt (falls vorhanden)
2. das eigentliche Archiv wird ausgepackt und in das Dateisystem kopiert
3. das Post-Install-Skript wird ausgeführt (falls vorhanden)

Ein ähnliches Schema wird bei der Deinstallation angewandt, auch hier gibt es häufig ein **Pre-Uninstall**- und **Post-Uninstall-Skript**.

Andere Distributoren, wie z. B. *SuSE* oder *Mandrake*, sind mittlerweile auch auf den RPM-Zug aufgesprungen, so dass dieses Format recht häufig im Internet anzutreffen ist. Allerdings kann man nicht einfach ein *SuSE* `rpm` unter *Mandrake* installieren oder umgekehrt, da die Pakete von den verschiedenen Distributoren teilweise unterschiedlich zusammengebaut werden.

Mit `rpm` kann man Pakete einzeln, aber auch mehrere auf einmal installieren, erneuern oder entfernen. Sind Pakete dabei, die voneinander abhängig sind, sortiert sie `rpm` in der richtigen Reihenfolge für die Installation. Dies bedeutet eine erhebliche Erleichterung für den Administrator, da er sich keine Gedanken darüber zu machen braucht, welche Pakete er zuerst installieren muss -- er gibt einfach alle in Frage kommenden Pakete an.

Kommando	Kurzbeschreibung
`rpm -ih x.rpm`	**Installation**; die Option `-h` (oder auch `-vh`) gibt zusätzlich noch einen Fortschrittsbalken aus
`rpm -U x.rpm`	**Update**; werden Konfigurationsdaten verändert, werden sie vorher unter der Endung .rpmsave gesichert.

	Alternativ wird die neue Version einer Konfigurationsdatei mit der Endung .rpmnew angelegt. Während des Updates macht der RedHat Package Manager auf diese Aktionen aufmerksam.
`rpm -qa`	**Query** -- Abfrage aller Pakete; ohne die Option `-a` kann man gezielt nach einem Paket nachfragen (z.B. `rpm -q fileutils`) Hilfreich ist auch die Option `-f`, mit der man abfragen kann, zu welchem Paket eine Datei (z. B. `/bin/ls`) gehört.
`rpm -e x.rpm`	**Erase** -- zum Deinstallieren eines Paketes
`rmp -V x`	`Verify` -- ist das Paket noch ordnungsgemäß installiert oder hat da etwa jemand dran manipuliert?

Die Manual-Page von `rpm` ist recht umfangreich, entsprechend dem Umfang dieses Kommandos. In der Tabelle sind deswegen nur die wichtigsten Befehle aufgelistet, um einen schnellen Einstieg zu ermöglichen. Tiefergehende Information sind über `man rpm` abrufbar. Eine sehr ausführliche Beschreibung der Möglichkeiten von `rpm` findet sich unter http://www.rpm.org/max-rpm/.

3.1 Kompilieren von Source-RPMs

Hat man ein Paket nur in Source-Form vorliegen (xxx.src.rpm), ist die Option `--rebuild` ganz hilfreich. Sie sorgt dafür, dass das Paket nach dem Auspacken auch gleich kompiliert wird. Während hierfür bei `RPM`-Versionen bis 4.0.X auch der Befehl `rpm` zuständig ist, gibt es seit der Version 4.1 den Befehl `rpmbuild`.

Das Kompilieren eines **Source-RPMs** auf dem eigenen Rechner hat auch den Vorteil, dass die Programme auf jeden Fall zu den installierten Bibliotheken passen.

Generell ist es empfehlenswert, diesen Kompilationsvorgang nicht als Benutzer root durchzuführen. Um als normaler Benutzer einen `rebuild` durchzuführen, muß als erstes eine Datei .rpmmacros im Homeverzeichnis angelegt werden:

```
user@linux ~/ $ cat ~/.rpmmacros

%_topdir /tmp/mirko-redhat

user@linux ~/ $
```

Nun müssen noch einige Verzeichnisse angelegt werden:

```
user@linux ~/ $ mkdir /tmp/mirko-redhat
user@linux ~/ $ mkdir /tmp/mirko-redhat/SPECS
user@linux ~/ $ mkdir /tmp/mirko-redhat/BUILD
user@linux ~/ $ mkdir /tmp/mirko-redhat/SOURCES
user@linux ~/ $ mkdir /tmp/mirko-redhat/RPMS
user@linux ~/ $ mkdir /tmp/mirko-redhat/RPMS/i386
user@linux ~/ $ mkdir /tmp/mirko-redhat/RPMS/i686
user@linux ~/ $ mkdir /tmp/mirko-redhat/RPMS/noarch
user@linux ~/ $ mkdir /tmp/mirko-redhat/SRPMS
```

oder in einem Einzeiler:

```
user@linux ~/ $ mkdir -p
/tmp/mirko-redhat/{RPMS/i386,RPMS/noarch,BUILD,SOURCES,SPECS,SRPMS}
```

Jetzt kann man ein vorhandenes **Source-RPM** einfach wie folgt kompilieren:

```
user@linux ~/ $ rpm --rebuild mod_auth_pam-1.0a-1.src.rpm
```

oder aber bei RPM-Versionen ab 4.1:

```
user@linux ~/ $ rpmbuild --rebuild mod_auth_pam-1.0a-1.src.rpm
```

Nach Ausführen des Befehls wird der Kompilationsvorgang durchgeführt:

* Die unter SOURCES abgelegten Quellen werden unterhalb von BUILD ausgepackt.
* Eventuell vorhandene Patches (Quelltext-Änderungen, die der Fehlerkorrektur oder dem Anpassen an das System dienen) verändern den Quelltext.
* Dann wird meistens automatisch der unter Die klassische Installation beschriebene Ablauf aus `./configure`, `make`, `make install` ausgeführt. Allerdings werden die Dateien hierbei temporär unter /var/tmp/PAKET-root installiert, da man als normaler Benutzer ja keine Zugriffsrechte auf die Standardverzeichnisse /usr, /etc usw. hat.
* Nun werden noch automatisch eventuell auftretende Abhängigkeiten aufgelöst.
* Die dem Programm zugehörigen Dateien werden komprimiert und in einem RPM zusammengefasst.

Am Ende findet sich dann unter RPMS/i386 das fertige RPM-Paket, welches man dann als root installieren kann.

3.2 Anfragen der RPM-Datenbank

Neben den eigentlichen Programm- oder Source-Dateien, die gepackt vorliegen, enthalten RPM-Dateien zusätzliche Informationen, welche bei der Installation in einer Datenbank gespeichert werden. So umfasst ein RPM zusätzlich eine kurze Beschreibung des Programmes, den Installationszeitpunkt, die Zeit zu dem es kompiliert wurde, eine Auflistung aller dem Programm zugehörigen Dateien nebst Informationen über die Größe dieser Dateien und einen **MD5-Hash**, durch den sich nachträglich überprüfen lässt, ob die Dateien geändert wurden.

Auch sind in einem RPM die Abhängigkeiten von anderen Bibliotheken abgespeichert, so dass das Aufspielen einer neuen, inkompatiblen Bibliotheksversion durch den RedHat Package Manager verhindert wird. Außerdem lassen sich in einer RPM-Datei Skripte unterbringen, die vor bzw. nach der Installation bzw. Deinstallation eines Programmes automatisch ausgeführt werden. Diese können dann z.B. einen Dienst automatisch als zu startendes Programm eintragen oder einen neuen Benutzer hinzufügen (bei Datenbanken, Web- und Mailservern gebräuchlich) bzw. diese Aktionen bei der Deinstallation rückgängig machen.

Die in der Datenbank während der Installation eingetragenen Informationen lassen sich jederzeit abfragen (s. Tabelle)

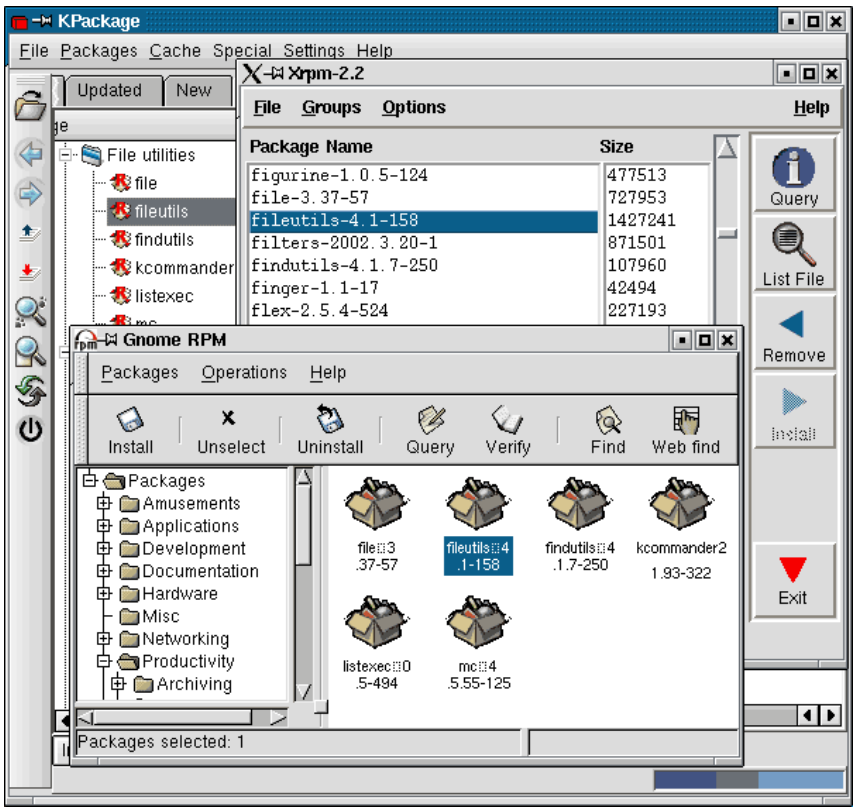
Option/Argument	Bedeutung	Beispiel
`-q` query = Abfrage,	ob ein Paket installiert ist	`rpm -q fileutils`
`-qa`	Anzeige aller installierten Pakete	
`-qf` Dateiname	zu welchem Paket gehört die Datei?	`rpm -qf /bin/ls => fileutils-4.1-4`
`-ql` Paketname	listet alle zum Paket gehörenden	`rpm -ql fileutils oder`

	Dateien	`rpm -qlf /bin/ls`
`-qi` Paketname	Infos zur Version, Inhaltsangabe, Installationsdatum, etc.	`rpm -qi fileutils`
`-qd` Paketname	zeigt nur die zum Paket gehörenden Dokumentationsdateien an	`rpm -qd xinetd`
`-qc` Paketname	zum Paket gehörende Konfigurationsdateien	`rpm -qc xinetd`
`-q --changelog` Paketname	Anzeigen des RPM-ChangeLog, dieses muss nicht gleichbedeutend mit dem der Software sein, da die Distributoren die Sourcen oft noch patchen.	`rpm -q --changelog openssl`

Viele dieser Abfrageoptionen lassen sich auch auf noch nicht installierte RPM-Pakete anwenden, hierzu dient die Option -p:

```
user@linux ~/ $ rpm -qip /mnt/cdrom/RedHat/RPMS/pinfo-0.5-1.i386.rpm
```

3.3 Graphische RPM-Frontends



gnorpm, kpackage und xrpm

Wer mit der Kommandozeile des `rpm`-Kommandos auf Kriegsfuß steht oder Probleme hat, sich die wichtigsten Optionen zu behalten, hat die Auswahl zwischen mehreren graphischen Frontends, die aber nicht alle Optionen von rpm abdecken.

`kpackage` ist bei *KDE* dabei und unterstützt Drag & Drop, d. h. man kann ein heruntergeladenes Paket aus dem Datei-Manager heraus in `kpackage` hineinschieben und fallen lassen. Es versteht auch das Debian-Paketformat, das an der Endung .deb erkennbar ist.

`GnoRPM` ist für Freunde des *Gnome*-Desktops.

`xrpm` ist ein in *Python* geschriebenes Frontend, das einfach zu bedienen ist und alle wichtigen Funktionen enthält.

`mc` -- der Midnight Commander ist zwar kein graphisches `RPM`-Frontend, kann aber `RPM`-Archive lesen und anzeigen

4 Debian Paket Format

Das Debian Paketformat ist detaillierter als RPM. *Debian* definiert nicht nur das Format, sondern auch die Datei-Struktur und vieles mehr. Deswegen ist das System problemloser aktualisierbar.

Während die meisten Distributionen inzwischen auf das RPM-Format umgestiegen sind, ist *Debian* seinem Paket-Format treu geblieben. Erkennbar sind diese Pakete an der Endung .deb. Zum Auspacken dient der *Debian* Packager (dpkg) oder das Kommando apt-get. dselect bietet ein Standard-Menü zur Paket-Installation, tasksel ein Menue mit verschiedenen vordefinierten Paketauswahlen. z.B. **x-window-system** oder **mail-server**.

Es gibt neben der Menü-gesteuerten Alternative (dselect, aptitude) auch graphische Frontends (gnome-apt, kpackage).

Kommando	**Beschreibung**
apt-get install <paketname>	Paket installieren
apt-get install <kernel-name>	anderen Kernel installieren
apt-get --purge remove <paketname>	Paket löschen
apt-get remove <paketname>	Paket löschen, aber Konfigurations-Dateien behalten
apt-get update / upgrade	System auf den neuesten Stand bringen

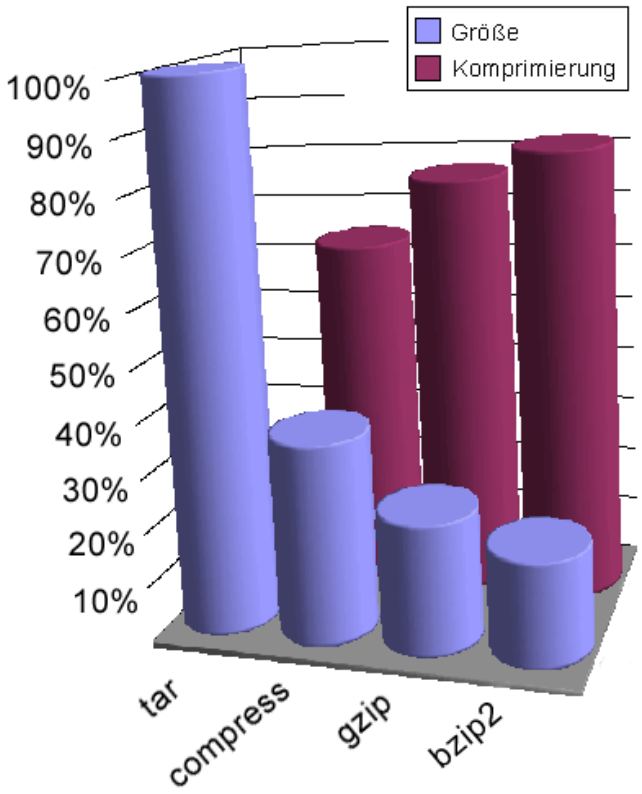
Ausführliche Informationen zu apt erhalten Sie in dem Text APT-Howto.

Da die Unterstützung von *Debian*-Paketen manchmal hinter der von RPM-Paketen hinterherhinkt, gibt es einen Konverter (alien), mit dem sich diese Pakete ins *Debian*-Format umwandeln lassen (und umgekehrt). Kritisch für eine Konvertierung sind systemnahen Paketen, da hier hierbei evtl. wichtige Informationen verloren gehen können.

Weitere Angaben zu *Debian* können dem Online-Manuel (man ...) oder dem *Debian* GNU/Linux Anwenderhandbuch (http://www.openoffice.de/linux/buch/) entnommen werden.

5 Die klassische Installation

Bevor Linux auf der Bildfläche erschien, wurden Programm-Pakete in Source-Form zur Verfügung gestellt, die in komprimierte **Tar**-Archive (auch als **Tar-Ball** bezeichnet) verpackt wurden. Während früher hauptsächlich das Unix-eigene compress zum Komprimieren verwendet wurde, ist es inzwischen weitgehend von gzip verdrängt worden, das einen besseren Komprimierungs-Faktor erzielt. Vereinzelt wird auch bzip2 eingesetzt (z. B. von http://www.blackdown.org), da es noch einen Tick besser ist (vgl. Abbildung "tar-archive.png"} -- hier wurde zum Vergleich die **Tar**-Datei von tkcvs 6.4 herangezogen)



Typische Komprimierung von compress, gzip und bzip2

Endung	komprimiert mit	auspacken mit
.tar	(ohne)	`tar xvf` ...
.tar.Z	`compress`	`tar Zxvf` ...
.tar.gz	`gzip`	`tar zxvf` ...
.tgz	`gzip`	`tar zxvf` ...
.tar.bz2	`bzip2`	`tar jxvf` ...

Das **GNU-tar-Kommando**, das üblicherweise bei allen Linux-Distributionen verwendet wird, kann mit komprimierten **Tar**-Archiven umgehen (s. Tabelle). Andere Unix-Systeme (z. B. SunOS) verwenden eine andere **Tar**-Implementierung. Hier muss man zuerst das Archiv dekomprimieren (mit `uncompress`, `gunzip` oder `bunzip2`), ehe man die **Tar**-Datei auspacken kann.

Vereinzelt findet man auch im Linux-Bereich **Zip**-Archive vor, erkennbar an der Endung .zip. Diese werden mit `unzip` ausgepackt.

Nachdem das **Tar**-Archiv erfolgreich ausgepackt ist, sollte man nach einer Datei README oder INSTALL Ausschau halten. Dort steht beschrieben, wie das Paket übersetzt und installiert wird. Unabhängig von der Plattform und Distribution sind es meist folgende Schritte, die ausgeführt werden:

1. `./configure` oder `make config`
   Im ersten Schritt wird untersucht, um was für ein System (Linux, Unix, ...) es sich handelt, welche

Bibliotheken vorhanden sind und ob die zur Kompilierung benötigten Tools wie C-Compiler (`gcc`) oder Linker (`ld`) installiert sind, um daraus ein Makefile zu generieren.

2. `make`
   Mit Hilfe des Makefiles, das im ersten Schritt erzeugt wurde, wird das Paket übersetzt.
3. `make test` (optional)
   Mit diesem Schritt wird überprüft, ob die Kompilation erfolgreich war.
4. `make install`
   Damit wird das Paket installiert.

Hilfreich bei der Übersetzung ist die Option `-n` des make-Kommandos. Damit kann man make erst einmal trocken ausführen, um zu sehen, welche Kommandos alle ausgeführt werden und in welches Verzeichnis welche Dateien kopiert werden, um nötigenfalls das Makefile noch anpassen zu können.

Auch wenn dieses Verfahren meist problemlos funktioniert, hat die Sache einen Haken: an die Deinstallation hat der Autor meistens nicht gedacht, d. h. ein `make uninstall` wird in den wenigsten Fällen klappen. Und so bleiben die installierten Dateien bis in alle Ewigkeit im System, es sei denn, man hat sich bei der Installation gemerkt, welche Dateien wohin kopiert wurden und löscht sie manuell.

Weitere Nachteile der manuellen Installation:

* Auf dem Zielsystem müssen alle Werkzeuge (Compiler, Linker, Make etc.), Bibliotheken und Headerdateien zum Kompilieren des Programmes vorhanden sein.
* Bei der Installation einer neueren Version eines Programmes (`Update`) werden evtl. die bereits vorhandenen, an das System angepassten Konfigurationsdateien der alten Version überschrieben.

6 Perl-Archive

Für Perl-Module gibt es als zentrale Anlaufstelle den **CPAN-Server** (Comprehensive Perl Archive Network, http://www.cpan.org), über den fast alle Perl-Module bezogen und direkt installiert werden können.

```
user@linux ~/ $ perl -MCPAN -e 'install Data::JavaScript'
```

Mit diesem Aufruf wird das **Data::JavaScript**-Modul installiert. Beim ersten Mal muss man evtl. noch die automatische Installation konfigurieren. Dazu wird man interaktiv durch verschiedene Fragen durchgelotst (z. B. wo das \cmd{gzip}- und \cmd{tar}-Kommando liegt, \dots).

Danach geht es mit der eigentlichen Installation los, bei der das angegebene Modul von einem **CPAN**-Server heruntergeladen, ausgepackt, getestet und installiert wird. War alles erfolgreich, sollte am Ende ein

```
/usr/bin/make install  -- OK
```

zu sehen sein. Falls nicht, kann es evtl. daran liegen, dass das angegebene Modul noch von weiteren Modulen abhängt, die nicht auf dem System vorhanden sind. In diesem Fall sollte man zuerst diese Module noch installieren.

7 Selbstauspackende Archive

In seltenen Fällen kommen auch Shell-Skripte zum Einsatz, die sich nach dem Aufruf selbst auspacken. Eventuell muss man vorher noch einige Fragen zur Installation beantworten. Meistens heißt das Skript install.sh und wird mit

```
user@linux ~/ $ ./install.sh
```

oder

```
user@linux ~/ $ sh install.sh
```

aufgerufen. Lässt sich das Skript nicht ausführen, empfiehlt es sich, die erste Zeile zu überprüfen. Sie sollte ein

Ausgabe nach der Eingabe von install.sh
`#!/bin/sh`

enthalten, was leider nicht immer der Fall ist.

8 Software-Archive

Linux ist Allgemeingut, dessen Bestandteile im Internet verstreut sind. Da es niemanden gehört, gibt es auch keine zentralen Stellen, die die ganzen Sourcen verwalten. Es gibt allerdings einige Anlaufstellen, von denen wir hier eine ganz kleine Auswahl präsentieren möchten (ohne Wertung):

* Sunsite
  Unter http://sunsite.unc.edu/pub finden sich neben GNU-Projekten auch andere OpenSource-Projekte und über 55 GB an Linux-Software und -Dokumentationen.
* Rpmfind
  http://rpmfind.net/linux/RPM ist ein riesiger Katalog von RPM-Archiven. Was man hier nicht findet, ist vermutlich auch nicht als RPM-Paket erhältlich.

Daneben gibt es natürlich noch die einzelnen Distributionen, die auch als Ausgangspunkt dienen können.